BhashaSetu: Cross-Lingual Knowledge Transfer from High-Resource to Low-Resource Language

Anonymous ACL submission

Abstract

Despite remarkable advances in natural language processing, developing effective systems for low-resource languages remains a formidable challenge, with performance typically lagging far behind high-resource counterparts due to data scarcity and insufficient linguistic resources. Cross-lingual knowledge transfer has emerged as a promising approach to address this challenge by leveraging resources from high-resource languages. In this paper, we investigate methods for transferring linguistic knowledge from high-resource languages to low-resource languages, where the number of labeled training instances is in hundreds. We focus on sentence-level and word-level tasks. We examine three approaches for cross-lingual knowledge transfer: (a) augmentation in hidden layers, (b) token embedding transfer through token translation, and (c) a novel method for sharing token embeddings at hidden layers using Graph Neural Networks. Experimental results on sentiment classification and NER tasks on low-resource languages Marathi, Bangla (Bengali) and Malayalam using high-resource languages Hindi and English demonstrate that our novel GNN-based approach significantly outperforms existing methods, achieving a significant improvement of 20 and 27 percentage points respectively in macro-F1 score compared to traditional transfer learning baselines such as multilingual and cross-lingual training. We also present a detailed analysis of the transfer mechanisms and identify key factors that contribute to successful knowledge transfer in this linguistic context. Our findings provide valuable insights for developing NLP systems for other low-resource languages.

1 Introduction

Cross-lingual knowledge transfer has emerged as a crucial approach for improving natural language processing capabilities across different languages. Recent advances in Large Language Models (LLMs) and multilingual model variants have demonstrated remarkable success in this domain by jointly training on multiple languages simultaneously, enabling zero-shot and few-shot learning capabilities. These models, such as XLM-R (Conneau et al., 2020), mT5 (Xue et al., 2021), and BLOOM (Scao et al., 2022), learn shared representations across languages, thereby facilitating knowledge transfer from high-resource to low-resource languages. The success of these models largely stems from their ability to leverage massive multilingual corpora and transformer-based architectures (Vaswani et al., 2017), which effectively capture cross-lingual patterns and relationships.

However, when dealing with extremely low-resource scenarios where target languages have very limited labeled data (e.g., only 100 training instances), even state-of-the-art multilingual models struggle (Wu and Dredze, 2020; Downey et al., 2024; Cassano et al., 2024) to generalize effectively. While parameter-efficient techniques like Adapter fine-tuning (Houlsby et al., 2019) and LoRA (Hu et al., 2022) reduce overfitting by updating fewer parameters, they still underperform in extreme low-resource settings. Cross-lingual models like AdaMergeX (Zhao et al., 2025) also fail to capture sufficient linguistic nuances with minimal target language examples. Most languages worldwide have extremely limited digital resources - India alone has at least 80 such languages, including several of its 22 official languages like Dogri, Bodo, and Kashmiri, while languages such as Saimar, Raji, and Toto have almost zero digital presence. For practical evaluation in this work, we use comparatively higher-resource languages (Marathi, Bangla, Malayalam) as simulated low-resource scenarios to enable rigorous testing with sufficient evaluation data.

To address this challenge, we propose a comprehensive framework that intelligently transfers linguistic knowledge from high-resource to low-resource languages through three complementary approaches. We name it **BhashaSetu** after the words "Bhasha" and "Setu" that mean "language" and "bridge" respectively in most Indian languages, highlighting its role in bridging languages.

Our approach is as follows. First, we introduce Hidden Augmentation Layers (HAL) that create mixed representations in the hidden space, allowing controlled knowledge transfer while preserving the target language's distinctive features. Second, we develop a token embedding transfer mechanism that leverages translation-based mappings to initialize low-resource language embeddings effectively. Finally, we propose a novel Graph-Enhanced Token Representation (GETR) approach that uses Graph Neural Networks (Zhou et al., 2020; Kipf and Welling, 2017; Veličković et al., 2018) to enable dynamic knowledge sharing between languages at the token level, thereby capturing complex cross-lingual relationships through graph-based message passing. This work contributes to the growing body of research in cross-lingual transfer learning while specifically addressing the challenges of extreme data scarcity in low-resource languages. In short, our contributions are:

1. We propose a comprehensive framework, BhashaSetu, for cross-lingual knowledge transfer in extremely low-resource scenarios, comprising three complementary approaches: hidden augmentation layer (HAL), token embedding transfer (TET), and a novel graph-enhanced token representation (GETR) with GNNs (Sec. 3).
2. We conduct extensive experiments across multiple NLP tasks (sentiment classification and NER) and language pairs spanning multiple languages, demonstrating the versatility and robustness of our approach. Experimental results on sentiment classification and NER tasks on low-resource languages Marathi, Bangla and Malayalam using high-resource languages Hindi and English demonstrate that our novel GNN-based approach significantly outperforms existing methods, achieving 20 and 27 percentage points improvement respectively in macro-F1 score compared to traditional baselines such multilingual joint training, LoRA and AdaMergeX while requiring only 100 training instances in the low-resource language (Sec. 4).
3. We provide systematic analysis of the impact of various factors on cross-lingual knowledge transfer, including mixing coefficient, architectural depth and dataset size ratios between languages.

2 Related Work

Cross-lingual transfer learning has advanced significantly with multilingual pre-trained models such as XLM-R (Conneau et al., 2020), mT5 (Xue et al., 2021), and PaLM (Chowdhery et al., 2022). While effective, these approaches require substantial multilingual training data, limiting their applicability in extreme low-resource settings. Recent parameter-efficient fine-tuning methods like LoRA (Hu et al., 2022), AdaMergeX (Zhao et al., 2025), and SALT (Lee et al., 2025) reduce overfitting risks but struggle with extremely limited target data.

Data augmentation techniques in hidden spaces, including mixup (Zhang et al., 2018; Verma et al., 2019) and their NLP adaptations (Chen et al., 2020; Sun et al., 2020), have proven valuable for low-resource scenarios and are comprehensively surveyed by (Feng et al., 2021). Token-level transfer approaches like trans-tokenization (Remy et al., 2024) enable cross-lingual embedding transfer without requiring parallel data, addressing a critical challenge for low-resource languages.

Graph-based cross-lingual methods such as Heterogeneous GNNs (Wang et al., 2021) depend on external semantic parsers and operate solely at the GNN level, without integrating graph knowledge into transformer models. Colexification-based multilingual graphs (Liu et al., 2023) construct graphs from colexification relations rather than token interactions, and similarly do not infuse graph information into transformers. While recent work has employed graph-based transformers with UCCA semantic graphs (Wan and Li, 2024), such approaches require pre-trained semantic parsers that are typically unavailable for low-resource Indian languages. In contrast, our GETR method constructs token-level graphs directly from training data and uniquely integrates GNN-based token interactions within the transformer, enabling dynamic, fine-grained cross-lingual knowledge sharing without external linguistic resources.

3 Methodology

This section presents three approaches for cross-lingual knowledge transfer: (a) augmentation in hidden layers, (b) token embedding transfer through translation, and (c) sharing token embeddings at hidden layers utilizing graph neural networks. Before delving into the technical details of these approaches, we first formally define the problem statement for cross-lingual knowledge transfer in low-resource scenarios.

3.1 Problem Statement

Let us formally define our notation for cross-lingual knowledge transfer. For a high-resource language, we denote the dataset of textual instances as $\mathbf{X_H} = \{x_1, x_2, \ldots, x_{N_H}\}$, where each x_i represents an individual text instance (e.g., a sentence). The corresponding task-specific outputs are represented as $Y_H = \{y_1, y_2, \ldots, y_{N_H}\}$, where N_H represents the total number of instances in the high-resource dataset, typically in the order of thousands. Similarly, we denote the low-resource language dataset as $\mathbf{X_L}$ and its corresponding outputs as Y_L, where $|\mathbf{X_L}| = N_L \ll N_H$, with N_L being extremely small (approximately 100 instances). This extreme data scarcity in the low-resource setting presents the core challenge in our task.

We define the combined dataset as $\mathbf{X} = \{\mathbf{X_H} \cup \mathbf{X_L}\}$ and $Y = \{Y_H \cup Y_L\}$. Our objective is to learn a model $\mathbf{M} : \mathbf{X} \to Y$ that maps input text instances from either or both $\mathbf{X_H}$ and $\mathbf{X_L}$ to their respective outputs, while effectively leveraging the high-resource language data to compensate for the limited low-resource samples. The output space Y can correspond to any encoder-based task, with two common task variants. The first is for sentence-level tasks (such as sentiment analysis) where $y_i \in \{0, 1, \ldots, c-1\}$, c being the number of classes. The second is for sequence-labeling tasks (such as NER): $y_i = [y_{i_1}, y_{i_2}, \ldots, y_{i_T}]$, where T is the sequence length and each token-level label $y_{it} \in \mathcal{Y}_{tags}$ represents a class (such as an NER tag).

Despite the different output structures, the core challenge of effective cross-lingual knowledge transfer remains consistent across tasks, allowing us to apply the same methodological approaches with task-specific adaptations. We next describe the three methods.

3.2 Augmentation in Hidden Layers (HAL)

Hidden layer augmentation has emerged as a prevalent technique for generating synthetic training data in the latent space when working with textual inputs (Zhang et al., 2018; Verma et al., 2019). While this approach has been successfully applied for domain adaptation within the same language (Zhang et al., 2022), its application to cross-lingual knowledge transfer, particularly from high-resource to low-resource languages, represents a novel direction. This method is particularly versatile as it can be applied to any high-resource and low-resource language pair, regardless of their script similarities or differences.

Let $\mathbf{E_M} : \mathbf{X} \to \mathbf{H}$ denote the encoder component of the model $\mathbf{M}$ that maps each input text x_i to its final encoded `CLS` representation $h^{\text{CLS}}i$. We propose a hidden augmentation mechanism that fuses knowledge from high-resource and low-resource languages through a weighted combination in the latent space. Formally, we generate new training pairs $A_i = (h^{\text{CLS}}_{A_i}, y_{A_i})$ as follows:

$$h^{\text{CLS}}_{A_i} = \alpha \cdot h^{\text{CLS}}_{H_i} + (1-\alpha) \cdot h^{\text{CLS}}_{L_i} \quad (1)$$

$$y_{A_i} = \begin{cases} \alpha \cdot y_{H_i} + (1-\alpha) \cdot y_{L_i}, \\ (\alpha \cdot y_{H_i,t} + (1-\alpha) \cdot y_{L_i,t})_{t=1}^{T} \end{cases} \quad (2)$$

where $\alpha \in [0, 1]$ is a mixing coefficient that controls the contribution of each language. This coefficient can be either fixed through training or randomly sampled per iteration. $y_{H_i}, y_{L_i} \in \mathbb{R}^c$ are typically one-hot encoded vectors for sentence tasks with c classes, and for sequence tasks, $y_{H_i,t}, y_{L_i,t} \in \mathbb{R}^{|\mathcal{Y}_{tags}|}$ represent the tag distribution at position t.

Empirically, α values between 0.1 and 0.4 yield optimal results, as they maintain the primary characteristics of the low-resource language while supplementing it with knowledge from the high-resource language. Since the augmentation produces soft labels, we employ KL-divergence loss (Cui et al., 2023) instead of standard cross-entropy loss (Zhong et al., 2023) for soft labels and cross-entropy for hard labels during training. This framework can be further extended by adding multiple transformer layers above E_M and performing augmentation at each layer's `CLS` output, thus enabling hierarchical knowledge fusion.

3.3 Token Embedding Transfer through Translation (TET)

Traditional approaches often initialize token embeddings for low-resource languages randomly, which can lead to suboptimal performance, especially when training data is scarce. We propose an initialization strategy that leverages token embeddings from a high-resource language through translation mapping (Remy et al., 2024). This approach

Algorithm 1 Token Embedding Transfer through Translation (TET)

```
1: $V_L \leftarrow$ Set of unique words from LRL corpus
2: for all $w_l \in V_L$ do                                ▷ For each LRL word
3:     $w_h \leftarrow \text{TranslateToHRL}(w_l)$
4:     $T_h \leftarrow \text{HRLTokenize}(w_h)$
5:     $T_l \leftarrow \text{LRLTokenize}(w_l)$
6:     $E_h \leftarrow \{\text{GetPretrainedEmbeddings}(t) | t \in T_h\}$    ▷ HRL token embeddings
7:     $e_{\text{avg}} \leftarrow \text{Mean}(E_h)$
8:     for all $t_l \in T_l$ do                            ▷ For each LRL token
9:         $P_{t_l} \leftarrow \emptyset$          ▷ Initialize projected embeddings set
10:        for all $w' \in V_L$ do                   ▷ Check all LRL words
11:            if $t_l \in \text{LRLTokenize}(w')$ then
12:                $P_{t_l} \leftarrow P_{t_l} \cup \{e_{\text{avg}}\}$
13:            end if
14:        end for
15:        $E_l[t_l] \leftarrow \text{Mean}(P_{t_l})$ ▷ Final embedding for LRL token
16:    end for
17: end for
18: return $E_l$                    ▷ Dictionary of LRL token embeddings
```

provides a more informed starting point for the embedding matrix of the low-resource language, enabling effective fine-tuning even with limited training samples. The core idea is to initialize the token embeddings of the low-resource language using the semantic information captured in the pre-trained embeddings of their translated counterparts in the high-resource language. While this method assumes the availability of word-level translations for the training data of the low-resource language, it does not require any pre-trained models or large corpora in the low-resource language. In our experiments, we used pymultidictionary (Pizarro, 2025) primarily to make our translation process seamless, faster and automated for the languages in our study, followed by manual verification of the translations to ensure accuracy. For extremely low-resource languages without dictionary support, we recommend manually translating the limited training vocabulary (which is manageable given the small dataset size of 100 instances).

Algorithm 1 details our systematic process for transferring token embeddings from a high-resource language (e.g., English) to a low-resource language (e.g., Marathi). To illustrate this process, consider transferring embeddings for the Marathi word "āntarbhāṣika" meaning "cross-lingual" in English. The word would be translated to English as "cross-lingual", which might be tokenized as "cross" + "lingual" in English. The word, "āntarbhāṣika" would be tokenized in Marathi, potentially splitting it into subword tokens like "āntar" + "bhāṣika". The pre-trained embeddings for these

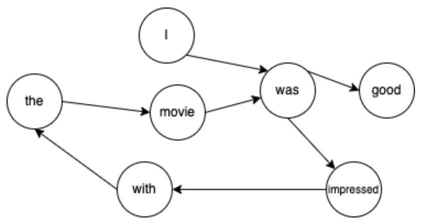

Figure 1: Graphical representation of tokens of two sentences in a batch: “The movie was good” and “I was impressed with the movie”.

English tokens are retrieved and averaged. For each Marathi token, we collect all instances where it appears across different words in the Marathi corpus. For example, the token "bhāṣika" might also appear in words like "bahubhāṣika" (meaning "multilingual"). Finally, we average all corresponding English embedding projections to create the final embedding for each Marathi token. While we show transliterated examples here for clarity, in our actual experiments we used the original scripts for all languages.

3.4 Graph-Enhanced Token Representation for Cross-lingual Learning (GETR)

We propose a novel approach leveraging Graph Neural Networks (GNN) (Zhou et al., 2020) to enable dynamic knowledge sharing between high-resource and low-resource languages at the token level. For each batch of mixed-language inputs, we construct an undirected graph $G = (T, C)$, where $T = \{t_1, t_2, \ldots, v_{N_k}\}$ represents the set of N unique tokens in batch k. The edge set C captures sequential relationships between tokens, defined as $C \subseteq \{t_{ij}, t_{i(j+1)} | t_{ij}, t_{i(j+1)} \in T\}$, where tokens $t_{i1}, t_{i2}, \ldots t_{in}$ form sentence s_i.

To illustrate the mechanism, consider two sentences: "The movie was good" from a high-resource language and "I was impressed with the movie" from a low-resource language. As shown in Figure 1, tokens are represented as nodes with edges connecting consecutive tokens within each sentence. When computing the representation for shared tokens (e.g., "was"), the model incorporates contextual information from both language environments. This allows the `CLS` embedding of the low-resource sentence to benefit from the high-resource language’s token representations through neighborhood aggregation.

Given the encoder output $\mathbf{H} \in \mathbb{R}^{B \times S \times D}$ (where B, S, and D denote batch size, sequence length, and embedding dimension respectively), we reshape it to $\mathbf{H}' \in \mathbb{R}^{L \times D}$ ($L = BS$) for GNN pro-

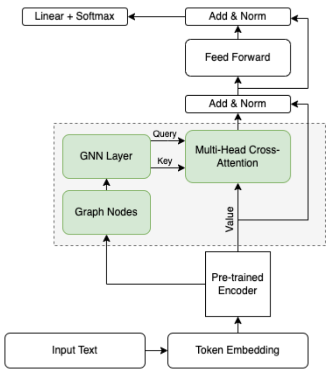

Figure 2: BERT encoder architecture incorporating the GNN layer for cross-lingual knowledge transfer.

cessing. We employ either GCN (Kipf and Welling, 2017) or GAT (Veličković et al., 2018) layers with an adjacency matrix $\mathbf{A} \in \{0,1\}^{L\times L}$ that captures token relationships such as $\mathbf{A}_{ij} = 1$ if l_i and l_j are consecutive tokens in a sentence. Notably, we construct $\mathbf{A}$ using the flattened dimension L rather than unique tokens, allowing for token repetition which makes the array multiplication simpler and straight-forward. The GNN output is then reshaped to generate query $\mathbf{Q}$ and key $\mathbf{K}$ matrices for the subsequent transformer layer, while the value $\mathbf{V}$ matrix maintains its original computation path:

$$\begin{aligned}\mathbf{H}' &= \text{Reshape}(\mathbf{H}) \in \mathbb{R}^{L\times D}\\ \mathbf{H}'_{\mathbf{G}} &= \text{GNN}(\mathbf{H}')\\ \mathbf{H}_{\mathbf{G}} &= \text{Reshape}(\mathbf{H}'_{\mathbf{G}}) \in \mathbb{R}^{B\times S\times D}\\ \mathbf{Q} &= \mathbf{H}_{\mathbf{G}} \times \mathbf{W}_{\mathbf{q}}\\ \mathbf{K} &= \mathbf{H}_{\mathbf{G}} \times \mathbf{W}_{\mathbf{k}}\end{aligned} \tag{3}$$

where $\mathbf{W_q} \in \mathbb{R}^{D\times D'}$, $\mathbf{W_k} \in \mathbb{R}^{D\times D'}$ are query and key weight matrices respectively. The subsequent transformer operations remain unchanged, following the standard sequence of cross-attention, feed-forward networks, layer normalization, and residual connections.

$$\mathbf{V} = \mathbf{H} \times \mathbf{W_v} \tag{4}$$

where $\mathbf{W_v} \in \mathbb{R}^{D\times D'}$ is the value weight matrix. Once $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are computed, the rest of the transformer encoder (Vaswani et al., 2017) block is unchanged, i.e., cross-attention block followed by feed-forward, layer normalization and residual connection. Figure 2 illustrates our modified BERT architecture with GNN layers (gray shaded area). Multiple GNN layers can be stacked sequentially to enable deeper cross-lingual knowledge transfer.

Strategic Batch Formation for Graph Construction: We propose a batch formation strategy that balances high-resource and low-resource instances while maximizing token overlap between languages. For every batch of size B, we ensure exactly $B/2$ instances from each language domain. Our construction alternates between low-resource and high-resource anchors: we first select a random low-resource instance, then add $(n/2-1)$ neighbors from low-resource language and $n/2$ from high-resource language based on maximum token overlap. These n instances are removed from the available pool to prevent repetition within an epoch. We then select a high-resource anchor and repeat the process, and continue this alternation until the batch is filled.

To improve robustness, 70% of the batches follow this strategic formation while the remaining 30% maintain an equal language distribution that selects instances randomly. This prevents over-reliance on specific token patterns while preserving structured knowledge transfer. The process continues across epochs until all low-resource instances are utilized.

During inference, we apply the same principle using training data to form neighborhoods for test instances based on token overlap. This balanced batch construction creates our token interaction graph $G = (T, C)$, enabling effective cross-lingual token relationships without requiring pre-trained resources for the low-resource language.

When the script is not shared, the edge between the nodes (words) is established if both the words convey the same meaning. In this way, connection is established between languages. For translation, we have used a combination of pymultidictionary (Pizarro, 2025) and manual effort similar to TET.

4 Experiments and Results

4.1 Dataset

Our experiments evaluate cross-lingual knowledge transfer across multiple languages and tasks. For sentiment classification, we employ two high-resource languages: Hindi (Yadav, 2023; Sawant, 2023) and English (Akanksha, 2023), each with 12,000 labeled instances. We use two low-resource target languages: Marathi (Pingle et al., 2023), which shares the Devanagari script with Hindi, and Bangla (Bengali) (Sazzed and Jayarathna, 2019), a language close to Hindi but with its own script. All sentiment classification datasets contain binary labels (positive and negative) with balanced class distributions.

The original Marathi dataset contained 12,113 training and 1,000 test instances. To simulate an extreme low-resource scenario, we created three distinct splits: a training set of 100 instances randomly sampled from the original training set, a validation set of 1,500 instances also from the original training set, and a test set of 2,000 instances by combining the original 1,000 test instances with 1,000 additional samples from the training set. We deliberately increased the test set size to evaluate robustness. Similarly, for Bengali, we created non-overlapping splits of 100 training, 1,500 validation, and 2,000 test instances. Throughout our experiments, we maintain the strict constraint that no pre-trained models or significant linguistic resources are available for the low-resource languages.

For Named Entity Recognition, we maintain English and Hindi as high-resource languages, with the English NER dataset (Jain, 2022) comprising 12,000 training instances (17 unique entity tags) and the Hindi dataset (Murthy et al., 2022) containing 12,084 training instances (13 unique entity tags). We apply our methods to two low-resource target languages: Marathi (Patil et al., 2022) with 100 training instances, 1,500 validation and 2,000 test instances (14 unique entity tags), and Malayalam (Mhaske et al., 2022) (that uses a completely different script from both Hindi and English) with 100 training instances, 1,500 validation instances, and 2,000 test instances (7 unique entity tags).

4.2 Implementation Details

Following our strict low-resource assumption, we first trained a tinyBERT (Jiao et al., 2019) model from scratch using only 100 instances of each low-resource language, including training new tokenizers. We established several baselines for comparison: (1) Joint Training (JT), which trains on high and low resource languages simultaneously, similar to multilingual models; (2) JT-HRLAdapLRL, which sequentially fine-tunes on high-resource data followed by adapter-based fine-tuning on low-resource data; (3) XLMFT-HRLAdapLRL, which applies the same sequential approach but starts from XLM-R (Conneau et al., 2020); (4) LoRA (Hu et al., 2022), which updates low-rank decomposition matrices instead of full weights; and (5) AdaMergeX (Zhao et al., 2025), which combines multiple adaptation methods. For adapter fine-tuning, we used a reduction factor of 16, increasing total parameters by only 1%. For LoRA and AdaMergeX, we followed the recommended parameter settings for encoder tasks from their original papers.

For our high-resource languages, we utilized `l3cube-pune/hindi-albert` (Joshi, 2022) for Hindi and `albert/albert-base-v2` (Lan et al., 2019) for English across both tasks. All experiments were conducted on an Amazon EC2 `p4de.24xlarge` instance with 8 NVIDIA A100 GPUs (80 GB each), using batch sizes of 128 for most approaches (8 for scratch training due to low number of data points, and 120 for GETR methods to accommodate graph construction using 9 neighbors per instance). We employed AdamW with learning rates between 3e-5 and 3e-7 for pre-trained models, and 3e-4 for scratch training with TET. Models were trained for 50 epochs with best checkpoints selected via validation loss.

For a fairer comparison between methods, we carefully balanced parameter counts across all models. Since GETR adds additional GNN layers to the architecture, we removed transformer layers from the pre-trained model to maintain comparable model size. For example, in GETR-GAT, we removed the last 3 transformer encoder layers and added 2 GAT layers, resulting in a parameter count (237,558,024) nearly identical to the Joint Training model (237,557,762). All experiments used the original scripts of the respective languages rather than transliteration. For baseline models, HAL, and GETR approaches, we leveraged pre-trained tokenizers from high-resource languages, augmenting them with new tokens from low-resource languages. These newly added tokens were randomly initialized, allowing the model to learn appropriate representations during training

All reported results are evaluated on test sets carefully selected to ensure no overlap with training data (Table 1). As expected, with such limited data and no pre-trained knowledge, Scratch Training models failed to learn meaningful patterns, defaulting to macro-F1 scores of 0.33 and 0.03 for sentiment classification and NER respectively.

4.3 Results on Sentiment Classification Task

Using English as the high-resource language, the baseline models achieve macro-F1 scores ranging from 0.53 to 0.55 for Marathi, with AdaMergeX performing best among them. Our proposed HAL method with $\alpha = 0.2$ and two layers shows improvement (0.63 with TET), but the GETR-GAT approaches demonstrate substantially greater gains, with GETR-GAT+HAL achieving the best performance (0.75 macro-F1), representing a 20 percentage point improvement over the best baseline. For Bangla as the low-resource language, baseline models achieve macro-F1 scores of 0.63, while GETR-GAT+HAL+TET delivers the best performance at 0.75, a 12 percentage point improvement.

With Hindi as the high-resource language, baseline performance improves significantly (up to 0.76 for AdaMergeX with Marathi), highlighting the benefit of script similarity. When Hindi is used as HRL and Marathi as LRL, TET is not required as they share the Devanagari script, ensuring that Marathi tokens already have pre-trained embeddings from the Hindi model. This explains the absence of TET-based results for this language pair in Table 1. Our HAL approach further boosts performance (0.80 macro-F1), while GETR-GAT+HAL achieves the highest score for Marathi (0.87 macro-F1), an 11 percentage point improvement over the best baseline. For Hindi-Bangla, GETR-GAT+HAL+TET reaches 0.81 macro-F1, outperforming the best baseline (AdaMergeX at 0.69) by 12 percentage points.

4.4 Results on NER Task

We extended our evaluation to Named Entity Recognition using test sets of approximately 2,000 instances for both Malayalam and Marathi. For Malayalam, the baseline models achieve macro-F1 scores between 0.26 and 0.28 with English as the high-resource language, while GETR-GAT+HAL+TET delivers 0.52, a substantial 24 percentage point improvement over the best baseline. With Hindi as the high-resource language, baseline models achieve similar scores (up to 0.28), while GETR-GAT+HAL+TET reaches 0.55, a 27 percentage point improvement. For Marathi NER, the pattern continues with GETR-GAT variants achieving macro-F1 scores of 0.40 with English (11 percentage points above the best baseline) and 0.44 with Hindi (6 percentage points above the best baseline HAL at 0.38).

These consistent improvements across different tasks and language families (Indo-Aryan and Dravidian) demonstrate that our GETR approach effectively transfers knowledge regardless of task type or target language. GETR's superior performance can be attributed to its ability to create dynamic, contextualized connections between tokens across languages, enabling more effective knowledge transfer at a granular level. Unlike static approaches, GETR allows low-resource language tokens to directly incorporate relevant semantic information from high-resource contexts through the graph structure, creating richer representations that better capture cross-lingual patterns. This transfer mechanism operates efficiently through the transformer's multi-head attention, where Q and K matrices capture the graph-based knowledge of tokens while preserving the original value computations, allowing cross-lingual information to propagate throughout the network. We measured inference time on an AWS p3.2xlarge instance across multiple datasets and found that our most complex GETR-GAT model (0.0101s ± 0.00001) with a neighbour size of 10 is only 6.31% slower than the Joint Training baseline (0.0095s ± 0.00001). We also observed that when using more complex approaches like HAL or GETR, TET's contribution diminishes.

We implemented additional baselines including HAL-LRL (augmentation within low-resource language only) and other three XLM-R finetuning variants, all performing comparably to Joint Training. We also evaluated GETR-GCN, but GETR-GAT consistently outperformed it due to GAT's adaptive edge weighting versus GCN's equal weighting of connections. Complete results for these experiments appear in Table 3 (appendix).

To evaluate the robustness of our approach and demonstrate its advantage over baseline methods, we compared BhashaSetu (our best-performing GETR-GAT+HAL configuration) with AdaMergeX (Zhao et al., 2025) across varying dataset sizes for NER with Hindi as HRL and Marathi as LRL (Table 2). The results reveal two critical insights. First, with extremely limited low-resource data (10-50 instances), AdaMergeX achieves modest performance (0.05-0.17 F1), while BhashaSetu demonstrates substantially better results even with minimal data, achieving 0.11 F1 with just 10 LRL instances and 0.34 F1 with 50 instances—representing a 17 percentage points improvement over AdaMergeX at these data scales. The fixed HRL size (12,000) experiment shows BhashaSetu's consistent advantage across all LRL sizes, with improvements of 9-17 percentage points, though the relative gap narrows as low-resource data increases.

The second exepriment, keeping LRL fixed at 100 instances while varying HRL size, reveals that AdaMergeX's performance degrades dramatically with decreasing HRL data (from 0.35 F1 with 12,000 instances to just 0.04 F1 with 500 in-

HRL	Method	Semantic Classification		NER	
		Marathi	Bangla	Marathi	Malayalam
-	Scratch Training	0.33 ±0.000 ($p<0.001$)	0.33 ±0.000 ($p<0.001$)	0.03 ±0.073 ($p<0.001$)	0.03 ±0.073 ($p<0.001$)
English	Joint Training	0.53 ±0.002 ($p<0.001$)	0.63 ±0.001 ($p<0.001$)	0.29 ±0.001 ($p<0.001$)	0.26 ±0.002 ($p<0.001$)
	Adapter Finetuning	0.53 ±0.001 ($p<0.001$)	0.63 ±0.002 ($p<0.001$)	0.29 ±0.001 ($p<0.001$)	0.26 ±0.001 ($p<0.001$)
	XLMFT-HRLAdapLRL	0.53 ±0.001 ($p<0.001$)	0.63 ±0.002 ($p<0.001$)	0.29 ±0.002 ($p<0.001$)	0.26 ±0.001 ($p<0.001$)
	LoRA	0.54 ±0.001 ($p<0.001$)	0.63 ±0.001 ($p<0.001$)	0.29 ±0.001 ($p<0.001$)	0.27 ±0.001 ($p<0.001$)
	AdaMergeX	0.55 ±0.001 ($p<0.001$)	0.63 ±0.001 ($p<0.001$)	0.29 ±0.001 ($p<0.001$)	0.28 ±0.002 ($p<0.001$)
	HAL	0.60 ±0.001 ($p<0.001$)	0.64 ±0.001 ($p<0.001$)	0.32 ±0.001 ($p<0.001$)	0.30 ±0.003 ($p<0.001$)
	HAL + TET	0.63 ±0.001 ($p<0.001$)	0.65 ±0.003 ($p<0.001$)	0.33 ±0.001 ($p<0.001$)	0.31 ±0.002 ($p<0.001$)
	GETR-GAT	0.73 ±0.001 ($p<0.001$)	0.72 ±0.001 ($p<0.001$)	**0.40** ±0.001 ($p=0.091$)	0.46 ±0.001 ($p<0.001$)
	GETR-GAT + TET	0.74 ±0.001 ($p<0.001$)	0.73 ±0.002 ($p<0.001$)	**0.40** ±0.001 ($p=0.114$)	0.47 ±0.003 ($p<0.001$)
	GETR-GAT + HAL	**0.75** ±0.001 (-)	0.74 ±0.001 ($p<0.001$)	**0.40** ±0.001 ($p=0.471$)	0.51 ±0.002 ($p<0.001$)
	GETR-GAT + HAL + TET	0.74 ±0.001 ($p<0.001$)	**0.75** ±0.001 (-)	**0.40** ±0.001 (-)	**0.52** ±0.001 (-)
Hindi	Joint Training	0.75 ±0.004 ($p<0.001$)	0.67 ±0.003 ($p<0.001$)	0.35 ±0.002 ($p<0.001$)	0.28 ±0.002 ($p<0.001$)
	Adapter Finetuning	0.74 ±0.002 ($p<0.001$)	0.67 ±0.002 ($p<0.001$)	0.34 ±0.003 ($p<0.001$)	0.28 ±0.002 ($p<0.001$)
	XLMFT-HRLAdapLRL	0.74 ±0.002 ($p<0.001$)	0.67 ±0.002 ($p<0.001$)	0.34 ±0.002 ($p<0.001$)	0.28 ±0.002 ($p<0.001$)
	LoRA	0.75 ±0.001 ($p<0.001$)	0.68 ±0.001 ($p<0.001$)	0.35 ±0.002 ($p<0.001$)	0.28 ±0.001 ($p<0.001$)
	AdaMergeX	0.76 ±0.001 ($p<0.001$)	0.69 ±0.001 ($p<0.001$)	0.36 ±0.001 ($p<0.001$)	0.28 ±0.001 ($p<0.001$)
	HAL	0.80 ±0.005 ($p<0.001$)	0.72 ±0.004 ($p<0.001$)	0.38 ±0.001 ($p<0.001$)	0.32 ±0.003 ($p<0.001$)
	HAL + TET	-	0.73 ±0.002 ($p<0.001$)	-	0.32 ±0.002 ($p<0.001$)
	GETR-GAT	0.85 ±0.001 ($p<0.001$)	0.79 ±0.002 ($p<0.001$)	**0.44** ±0.001 ($p=0.366$)	0.48 ±0.001 ($p<0.001$)
	GETR-GAT + TET	-	0.80 ±0.001 ($p<0.001$)	-	0.49 ±0.003 ($p<0.001$)
	GETR-GAT + HAL	**0.87** ±0.001 (-)	0.80 ±0.003 ($p<0.001$)	**0.44** ±0.001 (-)	0.53 ±0.002 ($p<0.001$)
	GETR-GAT + HAL + TET	-	**0.81** ±0.002 (-)	-	**0.55** ±0.001 (-)

Table 1: Performance (Macro-F1 score) comparison of different training approaches on sentiment classification and NER datasets when Hindi and English are considered as HRL and Marathi, Bangla and Malayalam as LRL. Values in parentheses are p-values from paired t-tests comparing each method against the best-performing approach for each HRL-LRL-task combination. The mean and standard deviation numbers are reported based on 5 independent runs.

LRL Size	HRL Size	Macro F1	
		AdaMergeX	BhashaSetu
10	12000	0.05 ± 0.001	0.11 ± 0.001
50	12000	0.17 ± 0.002	0.34 ± 0.002
100	12000	0.35 ± 0.001	0.44 ± 0.003
500	12000	0.39 ± 0.001	0.49 ± 0.002
1000	12000	0.42 ± 0.002	0.52 ± 0.001
5000	12000	0.55 ± 0.002	0.64 ± 0.003
10000	12000	0.71 ± 0.003	0.79 ± 0.002
100	12000	0.35 ± 0.001	0.44 ± 0.003
100	5000	0.22 ± 0.002	0.41 ± 0.002
100	1000	0.11 ± 0.028	0.25 ± 0.032
100	500	0.04 ± 0.025	0.10 ± 0.023

Table 2: NER performance comparison based on Macro-F1 between AdaMergeX and our approach (BhashaSetu) with Hindi as high-resource and Marathi as low-resource language under varying dataset sizes.

stances). While BhashaSetu also shows decreased performance with less HRL data, it maintains substantially better results (0.10 F1 even with just 500 HRL instances) and demonstrates greater resilience to HRL data reduction. These results highlight both BhashaSetu's effectiveness at enabling cross-lingual knowledge transfer and its superior ability to leverage limited high-resource data compared to AdaMergeX. Our additional experiments on sentiment classification (details in Tables 6 and 7 in appendix) reinforce these findings, with BhashaSetu outperforming AdaMergeX by 14-28 percentage points for Hindi-Bangla and 12-27 percentage points for English-Bangla pairs across various dataset sizes.

5 Conclusions

In this paper, we addressed the challenge of cross-lingual knowledge transfer for low-resource scenarios. We proposed three approaches: HAL, TET, and a novel GETR mechanism. Experimental results demonstrate that while traditional multilingual and cross-lingual models struggle with extreme data scarcity, our proposed approaches effectively leverage knowledge from high-resource languages.

Future work includes exploring self-supervised pre-training strategies specific to low-resource languages, more efficient graph construction algorithms, memory-optimized implementations of graph neural networks, and cross-lingual transfer for a wider range of tasks and language pairs.

Acknowledgements

Following ARR's AI Writing/Coding Assistance Policy, we acknowledge using Claude 3.7 Sonnet (Anthropic, 2025) for editorial assistance while maintaining full ownership of all scientific contributions.

Limitations

While our proposed approaches demonstrate strong performance across different tasks and language pairs, we acknowledge certain aspects that present opportunities for future research. Our experiments primarily focus on Indian languages from both Indo-Aryan and Dravidian families, which could be extended to typologically more distant language pairs with different word orders or morphological systems in future work.

Although BhashaSetu is effective with minimal low-resource data (100 instances), we observe that transfer performance correlates with high-resource language data availability, a common pattern in transfer learning approaches. This relationship between source data volume and transfer effectiveness presents an interesting direction for developing more data-efficient transfer techniques.

The Token Embedding Transfer approach benefits from word-level translation capabilities between language pairs. While such resources exist for many language combinations, future work could explore unsupervised methods for establishing cross-lingual correspondences when traditional bilingual dictionaries are unavailable.

Our Graph-Enhanced Token Representation approach introduces additional computational complexity during training and inference due to graph construction operations and GNN computations compared to simpler methods. However, this computational investment delivers substantially improved performance (21-27 percentage points gain in F1 scores), representing a favorable trade-off in many practical scenarios. Future implementations could explore optimization techniques to reduce this overhead.

Finally, while we demonstrate effectiveness on classification tasks (sentiment analysis and NER), extending these approaches to generative tasks involving neural machine translation or summary generation represents a promising direction for future research. This would further validate the versatility of our framework across the broader NLP task spectrum.

Ethics Statement

This research aims to promote linguistic inclusivity by addressing the technological disparity between high-resource and low-resource languages. We acknowledge that NLP capabilities have predominantly benefited widely-spoken languages, potentially exacerbating digital divides along linguistic lines. All datasets used in our experiments are publicly available with appropriate citations, and we did not collect or annotate new data that might introduce privacy concerns.

We recognize that transfer learning approaches may inadvertently propagate biases from source to target languages; however, our work takes a step toward mitigating representation disparities by enabling better performance with minimal labeled data in low-resource languages. Due to the focus on extremely low-resource settings (approximately 100 training instances), the computational requirements for target language adaptation were substantially lower than those typically needed for high-resource language model development, reducing the environmental impact compared to training large language models from scratch. While the GETR approaches do introduce additional computational overhead during the knowledge transfer process, the overall resource consumption remains modest relative to pre-training large multilingual models. This efficiency is particularly beneficial for researchers and practitioners with limited computational resources working on low-resource language technologies.

While we focused on Indian languages in this study, we believe that similar approaches could benefit other low-resource languages globally, contributing to more equitable language technology development. We emphasize that the performance improvements demonstrated should be considered within the context of the limitations described in our paper, and that practical applications would require careful consideration of cultural and linguistic nuances specific to each target community.

References

Akanksha. 2023. Sentiment analysis dataset. https://www.kaggle.com/code/akanksha10/sentiment-analysis-dataset. Accessed: 2024.

Anthropic. 2025. Claude 3.7 sonnet [large language model]. https://www.anthropic.com. Released February 24, 2025. Accessed 2025-05-19.

Federico Cassano, John Gouwar, Francesca Lucchetti, Claire Schlesinger, Anders Freeman, Carolyn Jane Anderson, Molly Q. Feldman, Michael Greenberg, Abhinav Jangda, and Arjun Guha. 2024. Knowledge transfer from high-resource to low-resource programming languages for code llms. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA2):Article 196, 1–32.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Jiequan Cui, Zhuotao Tian, Zhisheng Zhong, Xiaojuan Qi, Bei Yu, and Hanwang Zhang. 2023. Decoupled kullback-leibler divergence loss. *arXiv preprint arXiv:2305.13948*. Updated 2024.

C. M. Downey, Terra Blevins, Dhwani Serai, Dwija Parikh, and Shane Steinert-Threlkeld. 2024. Targeted multilingual adaptation for low-resource language families. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15647–15663, Miami, Florida, USA. Association for Computational Linguistics.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard H. Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*. Reviews various data augmentation techniques including neural augmentation and discusses augmentation applied in hidden representations.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Ben Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mary Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2790–2799.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Naman Jain. 2022. Ner dataset. `https://www.kaggle.com/datasets/namanj27/ner-dataset`. Retrieved May 17, 2025.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Raviraj Joshi. 2022. L3cube-hindbert and devbert: Pre-trained bert transformer models for devanagari based hindi and marathi languages. *arXiv preprint arXiv:2211.11418*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Seungyoon Lee, Seongtae Hong, Hyeonseok Moon, and Heuiseok Lim. 2025. Semantic aware linear transfer by recycling pre-trained language models for cross-lingual transfer. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16180–16193, Vienna, Austria. Association for Computational Linguistics.

Yihong Liu and 1 others. 2023. Crosslingual transfer learning for low-resource languages based on multilingual colexification graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1234–1245.

Arnav Mhaske, Harshit Kedia, Sumanth Doddapaneni, Mitesh M. Khapra, Pratyush Kumar, Rudra Murthy, and Anoop Kunchukuttan. 2022. Naamapadam: A large-scale named entity annotated data for indic languages. *arXiv preprint arXiv:2212.10168*.

Rudra Murthy, Pallab Bhattacharjee, Rishabh Sharnagat, Jyotsana Khatri, Diptesh Kanojia, and Pushpak Bhattacharyya. 2022. HiNER: A large hindi named entity recognition dataset. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4467–4476, Marseille, France. European Language Resources Association.

Parth Patil, Aparna Ranade, Maithili Sabane, Onkar Litake, and Raviraj Joshi. 2022. L3cube-mahaner: A marathi named entity recognition dataset and bert models. *arXiv preprint arXiv:2204.06029*.

Aabha Pingle, Aditya Vyawahare, Isha Joshi, Rahul Tangsali, and Raviraj Joshi. 2023. L3Cube-MahaSent-MD: A Multi-domain Marathi Sentiment Analysis Dataset and Transformer Models. *arXiv e-prints*, arXiv:2306.13888.

Pablo Pizarro. 2025. Pymultidictionary: Multilingual dictionary module for python. `https://github.com/ppizarror/PyMultiDictionary`. Version 5.1.0, Accessed: 2025-07-28.

François Remy, Pieter Delobelle, Hayastan Avetisyan, Alfiya Khabibullina, Miryam de Lhoneux, and Thomas Demeester. 2024. Trans-Tokenization and Cross-lingual Vocabulary Transfers: Language Adaptation of LLMs for Low-Resource NLP. *arXiv e-prints*, arXiv:2408.04303.

Onkar Sawant. 2023. Hindi sentiment analysis dataset. `https://www.kaggle.com/datasets/onkarsawant5613/hindi-sentiment-analysis`. Accessed: 2024.

Salim Sazzed and Sampath Jayarathna. 2019. A sentiment classification in bengali and machine translated english corpus. In *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 107–114. IEEE.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, and 1 others. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Lichao Sun, Yingce Xie, Ning Zhang, Ziyao Gao, Yan Xia, Tao Qin, Lijun Zhang, and Tie-Yan Liu. 2020. Mixup-transformer: Dynamic data augmentation for nlp tasks. In *COLING*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations (ICLR)*.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR.

Zhenhua Wan and Xiaofei Li. 2024. Exploring graph-based transformer encoder for low-resource neural machine translation. *PeerJ Computer Science*, 10:e1886.

Ziyun Wang, Xuan Liu, Peiji Yang, Shixing Liu, and Zhisheng Wang. 2021. Cross-lingual text classification with heterogeneous graph neural network. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 3096–3107.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual bert? In *Proceedings of the 5th Workshop on Representation Learning for NLP (RepL4NLP-2020)*, pages 120–130, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Siddharth Yadav. 2023. Hindi sentiment analysis. `https://github.com/sid573/Hindi_Sentiment_Analysis`.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Xinghua Zhang, Bowen Yu, Yubin Wang, Tingwen Liu, Taoyu Su, and Hongbo Xu. 2022. Exploring modular task decomposition in cross-domain named entity recognition. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 301–311, New York, NY, USA. Association for Computing Machinery.

Yiran Zhao, Wenxuan Zhang, Huiming Wang, Kenji Kawaguchi, and Lidong Bing. 2025. Adamergex: Cross-lingual transfer with large language models via adaptive adapter merging. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 9785–9800, Albuquerque, New Mexico. Association for Computational Linguistics.

Yutao Zhong and 1 others. 2023. Cross-entropy loss functions: Theoretical analysis and applications. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pages 23803–23828. PMLR.

Jie Zhou, Ganqu Cui, Shengding Zhang, Zhengyan Yang, Cheng Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

A Appendix

A.1 HAL

Figure 3 illustrates our modified architecture incorporating the hidden augmentation layer. The framework can be further extended by adding multiple transformer layers above E_M and performing augmentation at each layer's `CLS` output, thus enabling hierarchical knowledge fusion.

A.2 Results on Sentiment Classification Task and NER

We extensively evaluated our approach against multiple baselines, including parameter-efficient fine-tuning methods and XLM-R variants. For XLM-R, we tested: (1) XLMFT-LRL: fine-tuning only on low-resource data; (2) XLMFT-HRLRL: joint fine-tuning on both language datasets; (3) XLMFT-HRL2LRL: sequential fine-tuning on high-resource followed by low-resource data; and (4) XLMFT-HRLAdapLRL: fine-tuning on high-resource data followed by adapter-based fine-tuning on low-resource data with frozen base weights. Additionally, we evaluated LoRA and AdaMergeX as parameter-efficient alternatives, and HAL-LRL which applies augmentation only within the low-resource language. Our results show XLM-R variants perform comparably to Joint Training across

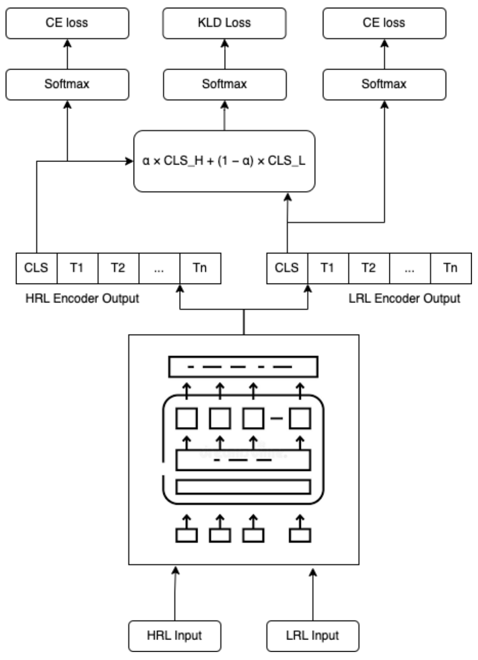

Figure 3: Architecture incorporating the Hidden Augmentation Layer (HRL and LRL inputs are high- and low-resource language inputs respectively)

all configurations, while HAL-LRL shows no improvement over Joint Training due to limited augmentation diversity in the extremely small low-resource dataset.

To understand the impact of mixing coefficient α in Hidden Augmentation Layer (HAL), we conducted experiments with different α values ranging from 0.1 to 0.8 (Table 4). For both English and Hindi as high-resource languages, α=0.2 yields the best performance, achieving accuracy/F1 scores of 0.610/0.590 and 0.860/0.860 respectively. The performance gradually degrades as α increases, with a more pronounced decline after α=0.5. This suggests that while knowledge from the high-resource language provides useful linguistic patterns and semantic structures, excessive reliance on it diminishes the model's ability to capture the unique characteristics and nuances of the low-resource language. The optimal performance at α=0.2 indicates that a balanced approach, where the model primarily learns from the low-resource language while leveraging complementary features from the high-resource language, is most effective. Notably, even with declining performance at higher α values, the model maintains reasonable performance (minimum accuracy of 0.590 for English and 0.830 for Hindi as HRL), indicating the robustness of the HAL approach across different mixing ratios.

We analyzed the impact of HAL depth by varying the number of layers from 1 to 6 (Table 5). For both English and Hindi as high-resource languages, 2 HAL layers yield optimal performance (accuracy/F1: 0.610/0.590 and 0.860/0.860 respectively), with secondary peaks at depth 4 for English (0.598/0.582) and depth 5 for Hindi (0.848/0.845), suggesting that while multiple HAL layers aid in knowledge transfer, excessive depth might lead to over-abstraction of features. Similarly, for both GETR-GCN and GETR-GAT approaches, three GNN layers demonstrated the best performance on the test set metrics, indicating an optimal depth for graph-based token interaction.

We extended our robustness evaluation to sentiment classification with Bangla as the low-resource language, testing both Hindi and English as high-resource languages (Table 6). The results reveal consistent advantages for BhashaSetu across all data configurations. With minimal low-resource data (10 instances), Joint Training achieves only 0.33 macro-F1 for both HRLs, while BhashaSetu reaches 0.61 with Hindi and 0.60 with English—an approximately 85% improvement. This advantage persists across all LRL sizes, though the gap narrows as training data increases. Hindi consistently outperforms English as the high-resource language, with BhashaSetu reaching 0.94 F1 using Hindi versus 0.89 F1 using English at 8,000 LRL instances.

The fixed LRL experiments (100 instances) with varying HRL size reveal BhashaSetu's remarkable resilience to limited high-resource data. With just 500 HRL instances, BhashaSetu maintains 0.62 F1 (Hindi) and 0.57 F1 (English), while Joint Training drops to 0.43 and 0.41 respectively. Most impressively, BhashaSetu with just 1,000 Hindi instances (0.73 F1) outperforms Joint Training with the full 12,000 instances (0.67 F1). These results demonstrate BhashaSetu's exceptional data efficiency in leveraging limited resources for cross-lingual transfer and confirm its effectiveness across both NER and sentiment classification tasks, regardless of the specific high-resource language used.

To evaluate the robustness of our approach on sentiment classification, we conducted extensive experiments varying dataset sizes with both Hindi and English as high-resource languages for Bangla (Table 6). With Hindi as HRL, BhashaSetu demonstrates remarkable effectiveness, achieving 0.61 macro-F1 with just 10 LRL instances compared to Joint Training's 0.33—an improvement of 28 percentage points. This advantage persists as LRL size increases, maintaining improvements of 12-

HRL	Method	Semantic Classification		NER	
		Marathi as LRL	**Bangla as LRL**	**Marathi as LRL**	**Malayalam as LRL**
-	Scratch Training	0.33 ± 0.000	0.33 ± 0.000	0.03 ± 0.073	0.03 ± 0.073
English	Joint Training	0.53 ± 0.002	0.63 ± 0.001	0.29 ± 0.001	0.26 ± 0.002
	Adapter Finetuning	0.53 ± 0.001	0.63 ± 0.002	0.29 ± 0.001	0.26 ± 0.001
	XLMFT-LRL	0.49 ± 0.002	0.60 ± 0.004	0.27 ± 0.005	0.23 ± 0.003
	XLMFT-HRLRL	0.53 ± 0.002	0.63 ± 0.001	0.29 ± 0.001	0.26 ± 0.001
	XLMFT-HRL2LRL	0.52 ± 0.002	0.63 ± 0.004	0.28 ± 0.003	0.24 ± 0.003
	XLMFT-HRLAdapLRL	0.53 ± 0.001	0.63 ± 0.002	0.29 ± 0.002	0.26 ± 0.001
	LoRA	0.54 ± 0.001	0.63 ± 0.001	0.29 ± 0.001	0.27 ± 0.001
	AdaMergeX	0.55 ± 0.001	0.63 ± 0.001	0.29 ± 0.001	0.28 ± 0.002
	HAL-LRL	0.52 ± 0.002	0.63 ± 0.001	0.29 ± 0.001	0.26 ± 0.001
	HAL	0.60 ± 0.001	0.64 ± 0.001	0.32 ± 0.001	0.30 ± 0.003
	HAL + TET	0.63 ± 0.001	0.65 ± 0.003	0.33 ± 0.001	0.31 ± 0.002
	GETR-GCN	0.69 ± 0.002	0.68 ± 0.001	0.36 ± 0.001	0.37 ± 0.001
	GETR-GCN + TET	0.68 ± 0.001	0.69 ± 0.003	0.36 ± 0.001	0.37 ± 0.002
	GETR-GCN + HAL	0.70 ± 0.001	0.70 ± 0.002	0.39 ± 0.001	0.43 ± 0.003
	GETR-GCN + HAL + TET	0.70 ± 0.002	0.70 ± 0.001	0.39 ± 0.001	0.43 ± 0.002
	GETR-GAT	0.73 ± 0.001	0.72 ± 0.001	0.40 ± 0.001	0.46 ± 0.001
	GETR-GAT + TET	0.74 ± 0.001	0.73 ± 0.002	0.40 ± 0.001	0.47 ± 0.003
	GETR-GAT + HAL	**0.75 ± 0.001**	0.74 ± 0.001	**0.40 ± 0.001**	0.51 ± 0.002
	GETR-GAT + HAL + TET	0.74 ± 0.001	**0.75 ± 0.001**	0.40 ± 0.001	**0.52 ± 0.001**
Hindi	Joint Training	0.75 ± 0.004	0.67 ± 0.003	0.35 ± 0.002	0.28 ± 0.002
	Adapter Finetuning	0.74 ± 0.002	0.67 ± 0.002	0.34 ± 0.003	0.28 ± 0.002
	XLMFT-LRL	0.71 ± 0.003	0.62 ± 0.005	0.30 ± 0.004	0.26 ± 0.004
	XLMFT-HRLRL	0.75 ± 0.001	0.67 ± 0.001	0.34 ± 0.001	0.28 ± 0.001
	XLMFT-HRL2LRL	0.75 ± 0.003	0.67 ± 0.004	0.34 ± 0.004	0.27 ± 0.001
	XLMFT-HRLAdapLRL	0.74 ± 0.002	0.67 ± 0.002	0.34 ± 0.003	0.28 ± 0.002
	LoRA	0.75 ± 0.001	0.68 ± 0.001	0.35 ± 0.002	0.28 ± 0.001
	AdaMergeX	0.76 ± 0.001	0.69 ± 0.001	0.36 ± 0.001	0.28 ± 0.001
	HAL-LRL	0.75 ± 0.003	0.67 ± 0.002	0.35 ± 0.001	0.27 ± 0.001
	HAL	0.80 ± 0.005	0.72 ± 0.004	0.38 ± 0.001	0.32 ± 0.003
	HAL + TET	-	0.73 ± 0.002	-	0.32 ± 0.002
	GETR-GCN	0.82 ± 0.001	0.75 ± 0.001	0.42 ± 0.002	0.38 ± 0.001
	GETR-GCN + TET	-	0.75 ± 0.002	-	0.38 ± 0.002
	GETR-GCN + HAL	0.83 ± 0.002	0.76 ± 0.001	0.43 ± 0.001	0.44 ± 0.003
	GETR-GCN + HAL + TET	-	0.76 ± 0.002	-	0.44 ± 0.002
	GETR-GAT	0.85 ± 0.001	0.79 ± 0.002	0.44 ± 0.001	0.48 ± 0.001
	GETR-GAT + TET	-	0.80 ± 0.001	-	0.49 ± 0.003
	GETR-GAT + HAL	**0.87 ± 0.001**	0.80 ± 0.003	**0.44 ± 0.001**	0.53 ± 0.002
	GETR-GAT + HAL + TET	-	**0.81 ± 0.002**	-	**0.55 ± 0.001**

Table 3: Performance (Macro-F1 score) comparison of different training approaches on sentiment classification and NER datasets when Hindi and English are considered as HRL and Marathi, Bangla and Malayalam as LRL. The mean and standard deviation numbers are reported based on 5 independent runs.

21 percentage points up to 8,000 instances (the maximum available in our Bangla dataset), where BhashaSetu achieves 0.94 macro-F1 compared to Joint Training's 0.82.

Similar patterns emerge with English as HRL, though with slightly lower absolute performance due to script differences. BhashaSetu achieves 0.60 macro-F1 with 10 LRL instances (27 percentage points over Joint Training) and maintains substantial improvements through 8,000 instances (0.89 vs 0.78 macro-F1). The fixed LRL experiments (100 instances) reveal BhashaSetu's superior resilience to HRL data reduction: with Hindi, performance drops from 0.81 to 0.62 macro-F1 as HRL size decreases from 12,000 to 500, while Joint Training falls more sharply from 0.67 to 0.43. English shows similar trends, with BhashaSetu maintaining better performance (0.75 to 0.57) compared to Joint Training's steeper decline (0.63 to 0.41). These results demonstrate BhashaSetu's effectiveness across different data regimes and language pairs, with particularly strong performance when languages share scripts.

Table 4: Performance comparison of HAL approach with different high-resource languages and varying α values. HRL: High Resource Language, LRL: Low Resource Language

HRL	LRL	α	Metrics	
			Accuracy	F1
English	Marathi	0.1	0.602±0.004	0.582±0.005
		0.2	**0.610±0.004**	**0.590±0.005**
		0.3	0.605±0.003	0.578±0.004
		0.4	0.598±0.004	0.571±0.005
		0.5	0.595±0.005	0.565±0.004
		0.6	0.592±0.004	0.558±0.005
		0.7	0.591±0.005	0.552±0.004
		0.8	0.590±0.004	0.550±0.005
Hindi	Marathi	0.1	0.852±0.004	0.848±0.005
		0.2	**0.860±0.003**	**0.860±0.005**
		0.3	0.848±0.004	0.845±0.004
		0.4	0.842±0.005	0.840±0.005
		0.5	0.838±0.004	0.835±0.004
		0.6	0.834±0.005	0.832±0.005
		0.7	0.832±0.004	0.831±0.004
		0.8	0.830±0.005	0.830±0.005

Table 5: Impact of HAL depth on model performance. HRL: High Resource Language, LRL: Low Resource Language

HRL	LRL	HAL Depth	Metrics	
			Accuracy	F1
English	Marathi	1	0.592±0.004	0.575±0.005
		2	**0.610±0.004**	**0.590±0.005**
		3	0.588±0.003	0.562±0.004
		4	0.598±0.004	0.582±0.005
		5	0.575±0.005	0.545±0.004
		6	0.570±0.004	0.540±0.005
Hindi	Marathi	1	0.842±0.004	0.838±0.005
		2	**0.860±0.003**	**0.860±0.005**
		3	0.835±0.004	0.832±0.004
		4	0.825±0.005	0.818±0.005
		5	0.848±0.004	0.845±0.004
		6	0.810±0.005	0.800±0.005

Table 6: Sentiment Classification performance comparison based on Macro-F1 between Joint Training (JT) and our approach (BhashaSetu) with Hindi and English as high-resource and Bangla as low-resource language under varying dataset sizes.

HRL	HRL Size	LRL Size	Macro F1 + JT	Macro F1 + BhashaSetu
Fixed HRL Size, Varying LRL Size				
Hindi	12000	10	0.33 ± 0.001	0.61 ± 0.001
Hindi	12000	50	0.51 ± 0.002	0.72 ± 0.002
Hindi	12000	100	0.67 ± 0.001	0.81 ± 0.003
Hindi	12000	500	0.69 ± 0.001	0.83 ± 0.002
Hindi	12000	1000	0.73 ± 0.002	0.87 ± 0.001
Hindi	12000	5000	0.79 ± 0.002	0.92 ± 0.003
Hindi	12000	8000	0.82 ± 0.003	0.94 ± 0.002
English	12000	10	0.33 ± 0.001	0.60 ± 0.001
English	12000	50	0.49 ± 0.002	0.68 ± 0.002
English	12000	100	0.63 ± 0.001	0.75 ± 0.003
English	12000	500	0.65 ± 0.001	0.78 ± 0.002
English	12000	1000	0.69 ± 0.002	0.81 ± 0.001
English	12000	5000	0.74 ± 0.002	0.87 ± 0.003
English	12000	8000	0.78 ± 0.003	0.89 ± 0.002
Fixed LRL Size, Varying HRL Size				
Hindi	12000	100	0.67 ± 0.001	0.81 ± 0.003
Hindi	5000	100	0.61 ± 0.002	0.76 ± 0.002
Hindi	1000	100	0.52 ± 0.023	0.73 ± 0.003
Hindi	500	100	0.43 ± 0.022	0.62 ± 0.006
English	12000	100	0.63 ± 0.001	0.75 ± 0.003
English	5000	100	0.55 ± 0.002	0.71 ± 0.002
English	1000	100	0.50 ± 0.023	0.65 ± 0.003
English	500	100	0.41 ± 0.022	0.57 ± 0.006

Table 7: NER performance comparison based on Macro-F1 between AdaMergeX and our approach (BhashaSetu) with English as high-resource and Marathi as low-resource language under varying dataset sizes.

LRL Size	HRL Size	Macro F1 + AdaMergeX	Macro F1 + BhashaSetu
Fixed HRL Size, Varying LRL Size			
10	12000	0.02 ± 0.001	0.11 ± 0.001
50	12000	0.13 ± 0.002	0.34 ± 0.002
100	12000	0.29 ± 0.001	0.40 ± 0.003
500	12000	0.34 ± 0.001	0.46 ± 0.002
1000	12000	0.39 ± 0.002	0.49 ± 0.001
5000	12000	0.51 ± 0.002	0.57 ± 0.002
10000	12000	0.64 ± 0.001	0.73 ± 0.001
Fixed LRL Size, Varying HRL Size			
100	12000	0.29 ± 0.001	0.40 ± 0.003
100	5000	0.18 ± 0.002	0.34 ± 0.002
100	1000	0.07 ± 0.025	0.20 ± 0.034
100	500	0.03 ± 0.022	0.07 ± 0.031